Sparse SOS Relaxations for Minimizing Functions that are Summations of Small Polynomials

Jiawang Nie[*] and James Demmel[†]

February 2, 2008

Abstract

This paper discusses how to find the global minimum of functions that are summations of small polynomials ("small" means involving a small number of variables). Some sparse sum of squares (SOS) techniques are proposed. We compare their computational complexity and lower bounds with prior SOS relaxations. Under certain conditions, we also discuss how to extract the global minimizers from these sparse relaxations. The proposed methods are especially useful in solving sparse polynomial system and nonlinear least squares problems. Numerical experiments are presented, which show that the proposed methods significantly improve the computational performance of prior methods for solving these problems. Lastly, we present applications of this sparsity technique in solving polynomial systems derived from nonlinear differential equations and sensor network localization.

Key words: Polynomials, sum of squares (SOS), sparsity, nonlinear least squares, polynomial system, nonlinear differential equations, sensor network localization

1 Introduction

Global optimization of multivariate polynomial functions contains quite a broad class of optimization problems. It has wide and important applications in science and engineering. Recently, there has been much work on globally minimizing polynomial functions using representation theorems from real algebraic geometry for positive polynomials. The basic idea is to approximate nonnegative polynomials by sum of squares (SOS) polynomials. This approximation is also called *SOS relaxation*, since not every nonnegative polynomial is SOS. Here a polynomial is said to be SOS if it can be written as a sum of squares of other polynomials. The advantage of SOS polynomials is that a polynomial is SOS if and only if a certain semidefinite program (SDP) formed by its coefficients is feasible. Since SDP [36] has efficient numerical methods, we can check whether a polynomial is SOS by solving a particular SDP.

To be more specific, suppose we wish to find the global minimum value f^* of a polynomial function $f(x)$ of vector $x = (x_1, x_2, \cdots, x_n) \in \mathbb{R}^n$. The SOS relaxation finds a lower bound γ for f^* such that the polynomial $f(x) - \gamma$ is SOS. Obviously, $f(x) - \gamma$ being SOS implies that $f(x) - \gamma$ is nonnegative for every real vector x. Hence such a γ is a lower bound. The maximum γ found this way is called the SOS lower bound, which is often denoted by f^*_{sos}. The relation $f^*_{sos} \leq f^*$ always holds for every polynomial $f(x)$ (it is possible that $f^*_{sos} = -\infty$). When $f^*_{sos} = f^*$, we say the SOS relaxation is *exact*. We refer to [16, 25, 26] for more details on SOS relaxations for polynomial optimization problems. There are two important issues for applying SOS relaxation in global optimization of polynomial functions: the *quality* and *computational complexity*.

---

[*]Department of Mathematics, University of California at San Diego, 9500 Gilman Drive, La Jolla, CA 92093. Email: njw@math.ucsd.edu

[†]Department of Mathematics and EECS, University of California, Berkeley, CA 94720. Email: demmel@cs.berkeley.edu

The quality means how good is the SOS lower bound f^*_{sos}. In practice, as observed in [25], many nonnegative polynomials that are not "artificially" constructed are SOS. However, Blekherman [5] pointed out that there are much more nonnegative polynomials than SOS polynomials. But usually SOS relaxation provides very good approximations, although theoretically it can fail with high probability. When SOS relaxation is not exact, i.e., $f^*_{sos} < f^*$, there are methods to fix it by applying modified SOS relaxations. Nie, Demmel and Sturmfels [23] proposed to use SOS representations of $f(x) - \gamma$ modulo the gradient ideal of $f(x)$, and show that the minimum value f^* can be obtained when f^* is attained at some point. Schweighofer [31] proposes to minimize $f(x)$ over a semialgebraic set called the *gradient tentacle*, and shows that the minimum value f^* can be computed when $f^* > -\infty$ but not attainable. Jibetean and Laurent [13], and Lasserre [17] propose perturbing $f(x)$ by adding a higher degree polynomial with tiny coefficients, and showed that the lower bounds will converge to the minimum value f^*. Recently, Laurent [19] gave a survey on solving polynomial optimization by using semidefinite relaxations. We refer to [13, 17, 19, 23, 31] for related work.

Another important issue for SOS relaxation is the computational complexity. Suppose $f(x)$ has degree $2d$ (it must be even for $f(x)$ to have a finite minimum). Then $f(x)$ has up to $\binom{n+2d}{2d}$ monomials. The condition that $f(x) - \gamma$ being SOS reduces to an SDP the size of whose linear matrix inequality (LMI) is $\binom{n+d}{d}$ with $\binom{n+2d}{2d}$ variables. These numbers can be huge for moderate n and d, say, $n = 2d = 10$. For large scale polynomial optimization problems, the general SOS relaxation is very difficult to implement numerically. Sometimes this complexity makes the applicability of SOS relaxation very limited. We refer to [25, 26] for the connection between SOS relaxation and SDP.

Prior work There is some work on exploiting sparsity in polynomial optimization when the polynomials are *sparse*. In such situations, sparse SOS relaxations are available and the resulting SDPs have reduced sizes, and hence larger problems can be solved. Here being sparse means that the number of monomials with nonzero coefficients is much smaller than the maximum possible number $\binom{n+2d}{2d}$. Kojima et al. [15] and Parrilo [27] discussed how to exploit sparsity of SOS relaxations in unconstrained polynomial optimization. Kim et al. [14] and Lasserre [18] discussed sparse SOS relaxations for constrained polynomial optimization problems and showed convergence under certain conditions. Waki et al. [34] proposed a heuristic procedure to exploit sparsity for minimizing polynomials by *chordal extension* of the *correlation sparsity pattern graph* (csp graph): the vertices of the csp graph are the variables $x_1, \cdots, x_n$; the edge (x_i, x_j) exists whenever $x_i x_j$ appears in one monomial of $f(x)$. To find one chordal extension, [34] proposed to use the symbolic sparse Cholesky factorization of the csp matrix with minimum degree ordering. If the chordal extension of the csp graph is also sparse, then the sparsity technique in [34] works well. However, if the chordal extension of the csp graph is much less sparse, then that sparsity technique might still be too expensive to be implementable for some practical problems.

Our contributions In many practical applications, the polynomials are not only sparse, but also given with certain sparsity patterns. For instance, the polynomials are often summations of other "small" polynomials, i.e. polynomials only involving a small number of variables. Sometimes, these representations contain useful information that might help us save computations significantly. These sparsity patterns are often ignored in prior work, where these polynomials would be treated using the usual "dense" algorithms. The main contribution of this paper is to propose new sparse SOS relaxation techniques taking the given sparsity pattern into account, and to show numerical experiments demonstrating their accuracy and speed.

In this paper, we consider the polynomial optimization problem of the form

$$\min_{x \in \mathbb{R}^n} \quad f(x) = \sum_{i=1}^{m} f_i(x_{\Delta_i}) \tag{1.1}$$

where $\Delta_i \subset [n] = \{1, 2, \cdots, n\}$. Here each $f_i(x_{\Delta_i})$ is a polynomial in $x_{\Delta_i} = (x_j | j \in \Delta_i)$. Let $\deg(f_i) = 2d_i$ and $2d = \deg(f) = \max\{2d_1, \cdots, 2d_m\}$ (we assume each f_i has even degree along with f). One basic and natural idea for solving problem (1.1) is to find the maximum γ such that

$$f(x) - \gamma = \sum_{i=1}^{m} s_i(x_{\Delta_i})$$

where each $s_i(x_{\Delta_i})$ is an SOS polynomial in x_{Δ_i} instead of all the variables $x_1, \cdots, x_n$. Exploiting this sparsity pattern can save significant computation without sacrificing much solution quality for many practical problems. In addition to presenting its numerical implementation, this paper will also discuss the theoretical properties of this sparse SOS relaxation and its variations.

The main distinction of our sparse SOS technique from earlier work like Waki et al. [34] is that we do not use the chordal extension of csp graphs. In case that the csp graph of $f(x)$ in (1.1) is chordal, our sparsity technique is almost the same as the one in [34]. However, if the csp graph of $f(x)$ is not chordal and its chordal extension is much less sparse, then our sparsity technique is significantly more efficient. If the csp graph of $f(x)$ is not chordal and its chordal extension is also sparse, then our sparsity technique is slightly more efficient while not losing much quality of solution. Furthermore, our sparsity technique can be applied to solve bigger dense polynomial optimization problems which can not be solved by other existing methods. This is due to the observation that every polynomial $g(x)$ is a summation of monomials whose number of variables is at most the degree $\deg(g)$. So, when $\deg(g)$ is small, like 4 or 6, then the formulation (1.1) is a good sparse model. The numerical computations show that our sparsity technique is usually more efficient than other existing methods in solving problems of the form (1.1).

Polynomial optimization problems of the form (1.1) have important practical applications: (i) *Solving polynomial systems:* Many large scale polynomial equations are often sparse and each equation might involve just a few variables, e.g., the polynomial equations obtained from discretization in nonlinear differential equations. Such polynomial systems can be equivalently transformed to a global polynomial optimization problems of the form (1.1). We will show that the proposed sparse SOS relaxation is exact when the polynomial system has at least one real solution. (ii) *Nonlinear least squares:* Many difficult problems in statistics, biology, engineering or other applications require solving certain nonlinear least squares problems and finding their global optimal solutions. If each equation is sparse, then sparse polynomial optimization (1.1) is a very natural model and our sparsity technique is very suitable. Sensor network localization is one important application of this kind.

Outline This paper is organized as follows. Section 2 introduces some notation and background for SOS relaxations, Section 3 discusses properties of the sparse SOS relaxation and its variations, Section 4 presents some numerical implementations, and Section 5 shows applications. Lastly, Section 6 draws some conclusions and discusses future work in this area.

2   Preliminaries

This section introduces some notations and backgrounds in SOS relaxation methods for minimizing polynomial functions.

Throughout this paper, we will use the following notation: $\mathbb{R}$ is the field of real numbers; $\mathbb{N}$ is the set of nonnegative integers; $\mathbb{R}^{\Delta_i} = \{(x_{k_1}, \cdots, x_{k_\ell}) : x_{k_j} \in \mathbb{R}\}$ when $\Delta_i = \{k_1, k_2, \cdots, k_\ell\}$; $\mathbb{R}[X]$: the ring of real polynomials in $X = (x_1, x_2, \cdots, x_n)$; $\mathbb{R}[X_{\Delta_i}]$: the ring of real polynomials in $X_{\Delta_i} = (x_k)_{k \in \Delta_i}$; $\sum \mathbb{R}[X]^2$: SOS polynomials in $\mathbb{R}[X]$; $\sum \mathbb{R}[X_{\Delta_i}]^2$: SOS polynomials in $\mathbb{R}[X_{\Delta_i}]$; $\sum \mathbb{R}_N[X]^2$: SOS polynomials in $\mathbb{R}[X]$ with degree at most $2N$; $\sum \mathbb{R}_N[X_{\Delta_i}]^2$: SOS polynomials in $\mathbb{R}[X_{\Delta_i}]$ with degree at most $2N$; $\|x\|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$; $x^\alpha = x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}$ for $\alpha \in \mathbb{N}^n$; $\operatorname{supp}(\alpha) = \{i \in [n] : \alpha_i \neq 0\}$; $\operatorname{supp}(f) = \{\alpha \in \mathbb{N}^n :$ the coefficient of x^α in $f(x)$ is nonzero$\}$; $|F|$ denotes the cardinality of set F; A^T denotes the transpose of matrix A; $A \succeq (\succ) 0$ means matrix A is positive semidefinite (definite); $\mathcal{M}_d(y)$ is the moment matrix of order d about $x \in \mathbb{R}^n$; $\mathcal{M}_d^{\Delta_i}(y)$ is the moment matrix of order d about $x_{\Delta_i} \in \mathbb{R}^{\Delta_i}$; $\mathcal{M}_F(y)$ is the moment matrix generated monomials with support F.

2.1. SOS and semidefinite programming (SDP)

A polynomial $p(x)$ in $x = (x_1, \cdots, x_n)$ is said to be sum of squares (SOS) if $p(x) = \sum_i p_i^2(x)$ for some polynomials $p_i(x)$. Obviously, if $p(x)$ is SOS, then $p(x)$ is *nonnegative*, i.e., $p(x) \geq 0$ for all $x \in \mathbb{R}^n$. However, the converse is not true. If $p(x)$ is nonnegative, then $p(x)$ is not necessarily SOS.

In other words, the set of SOS polynomials (which forms a cone) is properly contained in the set of nonnegative polynomials (which forms a larger cone). The process of approximating nonnegative polynomials by SOS polynomials is called SOS relaxation. For instance, the polynomial

$$x_1^4 + x_2^4 + x_3^4 + x_4^4 - 4x_1x_2x_3x_4$$
$$= \frac{1}{3} \left\{ (x_1^2 - x_2^2 - x_4^2 + x_3^2)^2 + (x_1^2 + x_2^2 - x_4^2 - x_3^2)^2 + (x_1^2 - x_2^2 - x_3^2 + x_4^2)^2 + \right.$$
$$\left. 2(x_1x_4 - x_2x_3)^2 + 2(x_1x_2 - x_3x_4)^2 + 2(x_1x_3 - x_2x_4)^2 \right\}$$

is SOS. This identity immediately implies that

$$x_1^4 + x_2^4 + x_3^4 + x_4^4 - 4x_1x_2x_3x_4 \geq 0, \ \ \forall (x_1, x_2, x_3, x_4) \in \mathbb{R}^4,$$

which is one arithmetic-geometric mean inequality.

The advantage of SOS polynomials over nonnegative polynomials is that it is more tractable to check whether a polynomial is SOS. To test whether a polynomial is SOS is equivalent to testing the feasibility of some SDP [25, 26], which has efficient numerical solvers. To illustrate this, suppose polynomial $p(x)$ has degree $2d$ (SOS polynomials must have even degree). Then $p(x)$ is SOS if and only if [25, 26] there exists a symmetric matrix $W \succeq 0$ such that

$$p(x) = \mathbf{m}_d(x)^T W \, \mathbf{m}_d(x)$$

where $\mathbf{m}_d(x)$ is the column vector of monomials up to degree d. For instance,

$$\mathbf{m}_2(x_1, x_2) = [\, 1, \ x_1, \ x_2, \ x_1^2, \ x_1x_2, \ x_2^2 \,]^T.$$

As is well-known, the number of monomials in x up to degree d is $\binom{n+d}{d}$. Thus the size of matrix W is $\binom{n+d}{d}$. This number can be very large. For instance, when $n = d = 10$, $\binom{n+d}{d} \geq 10^5$. However, for fixed d (e.g., $d = 2$), $\binom{n+d}{d}$ is polynomial in n. On the other hand, it is NP-hard (with respect to n) to tell whether a polynomial is nonnegative whenever $2d \geq 4$ (even when d is fixed)[16].

2.2. SOS relaxation in polynomial optimization

Let $f(x) = \sum_\alpha f_\alpha x^\alpha$ be a polynomial in x. Consider the global optimization problem

$$f^* := \min_{x \in \mathbb{R}^n} \ f(x).$$

This problem is NP-hard when $\deg(f) \geq 4$. The standard SOS relaxation is

$$f_{sos}^* := \max \quad \gamma$$
$$s.t. \quad f(x) - \gamma \text{ is SOS .}$$

Obviously we have that $f_{sos}^* \leq f^*$. In practice, SOS provides very good approximations, and often gives exact global minimum, i.e., $f_{sos}^* = f^*$, even though theoretically there are many more nonnegative polynomials than SOS polynomials [5].

In terms of SDP, the SOS relaxation can also be written as

$$f_{sos}^* := \max \quad \gamma \tag{2.1}$$
$$s.t. \quad f(x) - \gamma = \mathbf{m}_d(x)^T W \mathbf{m}_d(x) \tag{2.2}$$
$$W \succeq 0 \tag{2.3}$$

where $2d = \deg(f)$. The decision variable in the above is (γ, W) instead of x. The above program is convex about (γ, W). A lower bound f_{sos}^* can be computed by solving the resulting SDP. It can be shown [16] that the dual of (2.1)-(2.3) is

$$f_{mom}^* := \min_y \quad \sum_{|\alpha| \leq 2d} f_\alpha y_\alpha \tag{2.4}$$
$$s.t. \quad \mathcal{M}_d(y) \succeq 0 \tag{2.5}$$
$$y_{0,\cdots,0} = 1. \tag{2.6}$$

Here $\mathcal{M}_d(y)$ is the *moment matrix* generated by $y = (y_\alpha)$, a vector indexed by monomials of degree at most $2d$. The rows and columns of moment matrix $\mathcal{M}_d(y)$ are indexed by integer vectors. Each entry of $\mathcal{M}_d(y)$ is defined as

$$\mathcal{M}_d(y)(\alpha, \beta) := y_{\alpha + \beta}, \ \forall |\alpha|, |\beta| \leq d.$$

For instance, when $d = 2$ and $n = 2$, the vector

$$y = [\, y_{0,0}, \ y_{1,0}, \ y_{0,1}, \ y_{2,0}, \ y_{1,1}, \ y_{0,2}, \ y_{3,0}, \ y_{2,1}, \ y_{1,2}, \ y_{0,3}, \ y_{4,0}, \ y_{3,1}, \ y_{2,2}, \ y_{1,3}, \ y_{0,4} \,],$$

defines moment matrix

$$\mathcal{M}_2(y) = \begin{bmatrix} y_{0,0} & y_{1,0} & y_{0,1} & y_{2,0} & y_{1,1} & y_{0,2} \\ y_{1,0} & y_{2,0} & y_{1,1} & y_{3,0} & y_{2,1} & y_{1,2} \\ y_{0,1} & y_{1,1} & y_{0,2} & y_{2,1} & y_{1,2} & y_{0,3} \\ y_{2,0} & y_{3,0} & y_{2,1} & y_{4,0} & y_{3,1} & y_{2,2} \\ y_{1,1} & y_{2,1} & y_{1,2} & y_{3,1} & y_{2,2} & y_{1,3} \\ y_{0,2} & y_{1,2} & y_{0,3} & y_{2,2} & y_{1,3} & y_{0,4} \end{bmatrix}.$$

For SOS relaxation (2.1)-(2.3) and its dual problem (2.4)-(2.6), strong duality holds [16], i.e., their optimal values are equal ($f_{sos}^* = f_{mom}^*$). Hence f_{mom}^* is also a lower bound for the global minimum f^* of $f(x)$.

Now let us see how to extract minimizer(s) from optimal solutions to (2.4)-(2.6). Let y^* be one optimal solution. If moment matrix $\mathcal{M}_d(y^*)$ has rank one, then there exists one vector w such that $\mathcal{M}_d(y^*) = ww^T$. Normalize w so that $w_{(0, \cdots, 0)} = 1$. Set $x^* = w(2 : n+1)$. Then the relation $\mathcal{M}_d(y^*) = ww^T$ immediately implies that $y^* = \mathbf{m}_{2d}(x^*)$, i.e., $y_\alpha^* = (x^*)^\alpha$, so $f_{mom}^* = f(x^*)$. This says that a lower bound of $f(x)$ is attained at one point x^*. So x^* is one global minimizer.

When moment matrix $\mathcal{M}_d(y^*)$ has rank more than one, the process described above does not work. However, if $\mathcal{M}_d(y^*)$ satisfies the so-called *flat extension condition*

$$\operatorname{rank} \mathcal{M}_k(y^*) = \operatorname{rank} \mathcal{M}_{k+1}(y^*)$$

for some $0 \leq k \leq m - 1$, we can extract more than one minimizer (in this case the global solution is not unique). When the flat extension condition is met, it can be shown [8] that there exist distinct vectors $u_1, \cdots, u_r$ such that

$$\mathcal{M}_d(y^*) = \lambda_1 \mathbf{m}_d(u_1) \cdot \mathbf{m}_d(u_1)^T + \cdots + \lambda_r \mathbf{m}_d(u_r) \cdot \mathbf{m}_d(u_r)^T$$

for some $\lambda_i > 0$, $\sum_{i=1}^r \lambda_i = 1$. Here $r = \operatorname{rank} \mathcal{M}_d(y^*)$. The set $\{u_1, \cdots, u_r\}$ is called an *r-atomic representing support* for moment matrix $\mathcal{M}_d(y^*)$. All the vectors $u_1, \cdots, u_r$ can be shown to be global minimizers. They can be computed by solving some particular eigenvalue problem. We refer to [8] for flat extension conditions in moment problems and [12] for extracting minimizers.

2.3. Exploiting sparsity in SOS relaxation

As mentioned in the previous subsections, the size of matrix W in SOS relaxation is $\binom{n+d}{d}$, which can be very large. So SOS relaxation is expensive when either n or d is large. This is true for general dense polynomials. However, if $f(x)$ is sparse, i.e., its support $\mathcal{F} = \operatorname{supp}(f)$ is small, the size of the resulting SDP can be reduced significantly. Without loss of generality, assume $(0, \cdots, 0) \in \mathcal{F}$. Then $\operatorname{supp}(f) = \operatorname{supp}(f - \gamma)$ for any number γ.

Suppose $f(x) - \gamma = \sum_i \phi_i(x)^2$ is an SOS decomposition. Then by Theorem 1 in [29] we have

$$\operatorname{supp}(\phi_i) \subset \mathcal{F}^0 := \left(\text{ the convex hull of } \frac{1}{2} \mathcal{F}^e \right)$$

where $\mathcal{F}^e = \{\alpha \in \mathcal{F} : \ \alpha \text{ is an even integer vector}\}$. There exist some work [15, 34] on exploiting sparsity further. Here we briefly describe the technique introduced in [34].

For polynomial $f(x)$, define its *csp graph* $G = ([n], E)$ such that $(i, j) \in E$ if and only if $x_i x_j$ appears in some monomial of $f(x)$. Let $\{C_1, C_2, \cdots, C_K\}$ be the set of all maximal cliques of graph G. Waki et al. [34] proposed to represent $f(x) - \gamma$ as

$$f(x) - \gamma = \sum_{i=1}^{K} s_i(x), \quad \text{each } s_i(x) \text{ being SOS supp}(s_i) \subset C_i.$$

Theoretically, when $f(x) - \gamma$ is SOS, the above representation may not hold (see Example 3.5). And it is also difficult to find all the maximal cliques of graph G. Waki et al. [34] propose to replace $\{C_1, C_2, \cdots, C_K\}$ by the set of all maximal cliques of one *chordal extension* of G. We refer to [4] for properties of chordal graphs. For chordal graphs, there are efficient methods to find all the maximal cliques. Chordal extension is essentially the *sparse symbolic Cholesky factorization*. See [34] for more details on how to get the chordal extension.

We remark that in the worst case the sparse SOS relaxation above might be weaker than the general dense SOS relaxation even when the chordal extension is applied, as shown by Example 3.5.

There is much work in exploiting sparsity in SOS relaxations. We refer to [9, 14, 15, 27, 34] and the references therein.

3   The sparse SOS relaxation

Throughout this paper, we assume $f(x) = \sum_{i=1}^{m} f_i(x_{\Delta_i})$. Let $\|\Delta\|$ be the maximum cardinality of Δ_i, i.e., $\|\Delta\| = \max_i |\Delta_i|$. We are interested in the case that $\|\Delta\| \ll n$. To find the global minimum f^* of $f(x)$, we propose the following sparse SOS relaxation

$$f_\Delta^* := \max \quad \gamma$$
$$s.t. \quad f(x) - \gamma \in \sum_{i=1}^{m} \sum \mathbb{R}_d[x_{\Delta_i}]^2.$$

In terms of SDP, the above SOS relaxation is essentially the same as

$$f_\Delta^* := \max \quad \gamma \tag{3.1}$$
$$s.t. \quad f(x) - \gamma = \sum_{i=1}^{m} \mathbf{m}_d(x_{\Delta_i})^T W_i \mathbf{m}_d(x_{\Delta_i}) \tag{3.2}$$
$$W_i \succeq 0, \ i = 1, \cdots, m. \tag{3.3}$$

Notice that (3.2) is an identity. Let

$$\mathcal{F}_i = \{\alpha \in \mathbb{N}^n : \text{supp}(\alpha) \subset \Delta_i, |\alpha| \leq 2d\}, \quad \mathcal{F} = \bigcup \mathcal{F}_i. \tag{3.4}$$

Write $f(x) = \sum_\alpha f_\alpha x^\alpha$. Since $f(x) = \sum_i f_i(x_{\Delta_i})$, $f_\alpha \neq 0$ implies that $\alpha \in \mathcal{F}$. By comparing coefficients of both sides of (3.2), we have equality constraints

$$f_0 - \gamma = \sum_{i=1}^{m} W_i(0, 0), \quad f_\alpha = \sum_{i=1}^{m} \sum_{\eta + \tau = \alpha} W_i(\eta, \tau), \ \forall \alpha \neq 0. \tag{3.5}$$

Now we derive the dual problem for (3.1)-(3.3). Notice that constraint (3.2) is equivalent to the equality constraints (3.5). Let $y = (y_\alpha)_{\alpha \in \mathcal{F}}$ be the Lagrange multipliers for equations in (3.5), and U_i be the Lagrange multipliers for inequalities in (3.3). Each U_i is also positive semidefinite. The Lagrange function for problem (3.1)-(3.3) is

$$\mathcal{L} = \gamma + \left(f_0 - \gamma - \sum_i W_i(0, 0) \right) y_0 + \sum_{0 \neq \alpha \in \mathcal{F}} \left(f_\alpha - \sum_{i=1}^{m} \sum_{\eta + \tau = \alpha} W_i(\eta, \tau) \right) y_\alpha + \sum_i W_i \bullet U_i$$

$$= \gamma(1 - y_0) + \sum_{\alpha \in \mathcal{F}} f_\alpha y_\alpha + \sum_{i=1}^{m} \sum_{\alpha \in \mathcal{F}} \sum_{\eta + \tau = \alpha} W_i(\eta, \tau) \Big(U_i(\eta, \tau) - y_\alpha \Big).$$

So we can see that

$$\max_{\gamma, W_i} \mathcal{L}(\gamma, W_i, y_\alpha, U_i) = \begin{cases} \sum_\alpha f_\alpha y_\alpha & \text{if } y_0 = 1, U_i = \mathcal{M}_d^{\Delta_i}(y) \succeq 0 \, ; \\ +\infty & \text{otherwise.} \end{cases}$$

Therefore the dual of (3.1)-(3.3) is

$$f_\Sigma^* := \min \quad \sum_{\alpha \in \mathcal{F}} f_\alpha y_\alpha \tag{3.6}$$

$$\text{s.t.} \quad \mathcal{M}_d^{\Delta_i}(y) \succeq 0, \ i = 1, \cdots, m \tag{3.7}$$

$$y_0 = 1. \tag{3.8}$$

3.1. Complexity comparison

Since the dual of the standard or sparse SOS relaxation not only returns the SOS lower bound but also provides the moment matrix to help extract minimizers, we compare the computational complexity of (2.4)-(2.6) and (3.6)-(3.8). The LMI (2.5) is of size $\binom{n+d}{d} = \mathcal{O}(n^d)$ and has $\binom{n+2d}{2d} = \mathcal{O}(n^{2d})$ decision variables. At each step of an interior-point method (e.g., the dual scaling method [3]), the complexity for solving (2.4)-(2.6) is $\mathcal{O}(n^{6d})$. On the other hand, (3.7) has m LMIs, which are of sizes at most $\binom{\|\Delta\|+d}{d} = \mathcal{O}(\|\Delta\|^d)$, and $\mathcal{O}\left(m\binom{\|\Delta\|+2d}{2d}\right) = \mathcal{O}(m\|\Delta\|^{2d})$ decision variables. At each step of interior-point methods, the complexity for solving (3.6)-(3.8) is $\mathcal{O}(m^3\|\Delta\|^{6d})$. When $\|\Delta\|$ is independent of n and $m = \mathcal{O}(n^p)$ with $p < 2d$, then

$$\mathcal{O}(m^3\|\Delta\|^{6d}) \ll \mathcal{O}(n^{6d}).$$

Therefore (3.6)-(3.8) is much easier to solve than (2.4)-(2.6).

The complexity of sparse SOS relaxation in [34] depends on the chordal extension of the csp graph. In the worst case, it can be as big as for the general SOS relaxation (2.4)-(2.6). Let Ω be the maximum size of the maximal cliques of the chordal extension. In practice, Ω is often bigger than or equal to $\|\Delta\|$. When $\Omega > \|\Delta\|$, the SOS relaxation (3.6)-(3.8) is usually more efficient.

3.2. Lower bound analysis

Recall that $\mathcal{F}_i = \{\alpha \in \mathbb{N}^n : \text{supp}(\alpha) \subset \Delta_i, |\alpha| \leq 2d\}$. From the representation (1.1) of $f(x)$, we have

$$\text{supp}(f) \subseteq \bigcup_{i=1}^m \mathcal{F}_i.$$

This leads us to think that the relaxation (3.6)-(3.8) should give reasonable lower bounds, although it might be weaker than the general SOS (see Example 3.5).

Theorem 3.1. *The optimal values $f_\Sigma^*, f_\Delta^*, f_{sos}^*, f^*$ satisfy the relationship*

$$f_\Sigma^* = f_\Delta^* \leq f_{sos}^* \leq f^*.$$

Proof. The latter two inequalities are obvious because the feasible region defined by (3.7)-(3.8) contains the one defined by (2.5)-(2.6). To prove the first equality, by the standard duality argument for convex program, it suffices to show that (3.7) admits a strict interior point. Define $\hat{y} = (\hat{y}_\alpha)_{\alpha \in \mathcal{F}}$ as

$$\hat{y}_\alpha := \frac{\int_{\mathbb{R}^n} x^\alpha e^{-\|x\|_2^2} \, dx}{\int_{\mathbb{R}^n} e^{-\|x\|_2^2} \, dx}.$$

For every nonzero vector $\xi = (\xi_\alpha)_{\alpha \in \mathcal{F}_i}$, we have

$$\xi^T M_d^{\Delta_i}(\hat{y}) \, \xi = \frac{\int_{\mathbb{R}^n} \left(\sum_{|\alpha| \leq d} \xi_\alpha x^\alpha\right)^2 e^{-\|x\|_2^2} dx}{\int_{\mathbb{R}^n} e^{-\|x\|_2^2} dx} > 0.$$

So $M_d^{\Delta_i}(\hat{y}) \succ 0$ for every $1 \leq i \leq m$. Therefore $\hat{y}$ is an interior point for (3.6)-(3.8), which implies the strong duality $f_\Sigma^* = f_\Delta^*$. $\square$

Remark 3.2. Theorem 3.1 implies that the lower bound f_Δ^* given by (3.1)-(3.3) is weaker than the SOS lower bound f_{sos}^*. There are examples such that $f_\Delta^* < f_{sos}^*$ (see Example 3.5). However, in many numerical simulations, the lower bound f_Δ^* is very useful. For randomly generated polynomials, as shown in Section 4, it frequently happens that $f_\Delta^* = f_{sos}^*$. On the other hand, under some conditions, we can prove $f_\Delta^* = f_{sos}^*$.

Suppose $\Delta_1, \Delta_2, \cdots, \Delta_m$ satisfy the *running intersection property*:

$$\text{For every } 1 \le i \le m-1, \ \exists\, k \le i \ \text{ such that } \quad \Delta_{i+1} \cap \left(\bigcup_{j=1}^i \Delta_j \right) \subsetneq \Delta_k. \tag{3.9}$$

Theorem 3.3. *Suppose (3.6)-(3.8) has a optimal solution y^* such that each $\mathcal{M}_{d_i}^{\Delta_i}(y^*)$ has a representing measure μ_i on $\mathbb{R}^{\Delta_i}$. If condition (3.9) holds, then $f_\Delta^* = f_{sos}^*$.*

Proof. For any Δ_i, Δ_j, $\mathcal{M}_d^{\Delta_i \cap \Delta_j}(y^*)$ is a common principle submatrix of $\mathcal{M}_d^{\Delta_i}(y^*)$ and $\mathcal{M}_d^{\Delta_j}(y^*)$. So the marginals of measures μ_i are consistent, i.e., the restrictions of these measures on the common subspaces are the same. By Lemma 6.4 in [18], there exists a measure on $\mathbb{R}^n$ such that μ_i is the marginal of μ with respect to Δ_i for all $i = 1, \cdots, m$. Define vector $\tilde{y}$ such that

$$\mathcal{M}_d(\tilde{y}) = \int_{\mathbb{R}^n} \mathbf{m}_d(x)\mathbf{m}_d(x)^T \mu(dx).$$

Then every $\mathcal{M}_d^{\Delta_i}(y^*)$ is a principle submatrix of $\mathcal{M}_d(\tilde{y})$. So $\tilde{y}_\alpha = y_\alpha^*$ whenever $\mathrm{supp}(\alpha) \subset \Delta_j$ for some j. Since the $f_\alpha \ne 0$ implies $\mathrm{supp}(\alpha) \subset \Delta_j$ for some j, we know the objective value of (3.6) is the same for y^* and $\tilde{y}$. Thus $f_{sos}^* \le f_\Delta^*$. Since $f_{sos}^* \ge f_\Delta^*$, we get $f_{sos}^* = f_\Delta^*$. $\square$

Remark 3.4. The running intersection property (3.9) alone is not sufficient to guarantee the equality $f_\Delta^* = f_{sos}^*$, as shown by the following example.

Example 3.5. $f(x) = f_1(x_1, x_2) + f_2(x_2, x_3)$ where $f_1 = x_1^4 + (x_1 x_2 - 1)^2$ and $f_2 = x_2^2 x_3^2 + (x_3^2 - 1)^2$. Solving dense SOS relaxation (2.1)-(2.3) and sparse SOS relaxation (3.1)-(3.3) numerically, we find that

$$f_\Delta^* \approx 5.0 \cdot 10^{-5} \ < \ f_{sos}^* \approx 0.8499.$$

Actually the minimum $f^* \approx 0.8650$. First, solve equation $\nabla f(x) = 0$, and evaluate $f(x)$ on these critical points, then we find the minimum of these critical values is about 0.8650. So $f^* < 1$. Second, we prove that the minimum f^* is attainable. Let $\{x^{(k)}\}$ be a sequence such that $f(x^{(k)}) \to f^*$ as k goes to infinity. We claim that the sequence $\{x^{(k)}\}$ must be bounded.

Otherwise, suppose $x^{(k)} \to \infty$. Thus at least one of coordinates $x_1^{(k)}, x_2^{(k)}, x_3^{(k)}$ should go to infinity. If either $x_1^{(k)}$ or $x_3^{(k)}$ goes to infinity, then $f(x^{(k)})$ goes to infinity, which is not possible. So $x_2^{(k)} \to \infty$. Since $\{f(x^{(k)})\}$ is bounded, without loss of generality, we assume $x_1^{(k)} \to a_1$, $x_1^{(k)} x_2^{(k)} \to a_{12}$, $x_2^{(k)} x_3^{(k)} \to a_{23}$, $x_3^{(k)} \to a_3$ for some numbers a_1, a_{12}, a_{23}, a_3. If $a_3 = 1$, then $x_2^{(k)}$ is convergent to a_2, which is not possible. And, if $a_3 \ne 0$, then $x_2^{(k)} x_3^{(k)}$ goes to infinity, which is also not possible. So $a_3 = 0$, and hence

$$f(x^{(k)}) \ge (x_3^2 - 1)^2 \to 1 > f^*,$$

which is a contradiction.

So the sequence $\{x^{(k)}\}$ is bounded and has an accumulation point x^*. Then we must have $f(x^*) = f^*$, which means that f^* is attained at some point. From the computation of critical values, we know $f^* \approx 0.8650$. For this polynomial, both the dense and sparse SOS relaxation are not exact: $f_\Delta^* < f_{sos}^* < f^*$, and the method in [34] gives the same lower bound f_Δ^*.

Corollary 3.6. *If all f_i are quadratic and condition (3.9) holds, then $f_{sos}^* = f_\Delta^*$.*

Proof. When all f_i are quadratic, i.e., $d_i = 1$, the entries of moment matrix $\mathcal{M}_1^{\Delta_i}$ are the first and second order moments. The positive semidefiniteness of $\mathcal{M}_1^{\Delta_i}$ implies $\mathcal{M}_1^{\Delta_i}$ has a representing measure. Then the conclusion is immediately implied by Theorem 3.3. $\square$

Remark 3.7. If the running intersection condition (3.9) fails, then Corollary 3.6 is no longer true, as shown by the example below.

Example 3.8. Consider the polynomial $f(x) = f_1(x_1, x_2) + f_2(x_2, x_3) + f_3(x_1, x_3)$ where $f_1 = \frac{1}{2}(x_1^2 + x_2^2) + 2x_1x_2$, $f_2 = \frac{1}{2}(x_2^2 + x_3^2) + 2x_2x_3$, $f_3 = \frac{1}{2}(x_1^2 + x_3^2) + 2x_1x_3$. In this case

$$\Delta_1 = \{1, 2\}, \quad \Delta_2 = \{2, 3\}, \quad \Delta_3 = \{1, 3\}.$$

The running intersection property (3.9) fails. But we have $f_\Delta^* = -\infty < f_{sos}^* = f^* = 0$.

3.3. Extraction of minimizers

In this subsection, we discuss how to extract minimizer(s) $x^* = (x_1^*, \cdots, x_n^*)$. Suppose $y^* = (y_\alpha^*)_{\alpha \in \mathcal{F}}$ is one optimal solution to (3.6)-(3.8). Let $\delta_i = \{i\}$ for every i. The entries of y^* whose indices are supported in δ_i are

$$y_0^*, \ y_{e_i}^*, \ y_{2e_i}^*, \ \cdots, \ y_{2de_i}^*$$

which are the entries the moment matrix $M_d^{\delta_i}(y^*)$. So coordinate x_i^* can be extracted from moment matrix $M_d^{\delta_i}(y^*)$ if it satisfies the flat extension condition. Let $\mathcal{V}_i$ be the set of all the points that can be extracted from the moment matrix $M_d^{\delta_i}(y^*)$. If $\mathcal{V}_i$ is a singleton, then x_i^* has a unique choice.

The situation is more subtle if some $\mathcal{V}_i$ has cardinality greater than one. Suppose for some $i, j \in [n]$ we have $|\mathcal{V}_i| > 1$ and $|\mathcal{V}_j| > 1$. Can $x_i^* \ x_j^*$ appear simultaneously in the optimal solution x^* for arbitrarily chosen $x_i^* \in \mathcal{V}_i$, $x_j^* \in \mathcal{V}_j$? The answer is obviously no! For instance, the polynomial

$$(x_1^2 - 1)^2 + (x_2^2 - 1)^2 + (x_1 - x_2)^2$$

has only two global minimizers $\pm(1, 1)$. We find that $\mathcal{V}_1 = \mathcal{V}_2 = \{1, -1\}$. But obviously $(1, -1)$ and $(-1, 1)$ are not global minimizers.

Now what is the rule for matching x_i^* and x_j^* if $|\mathcal{V}_i| > 1$ or $|\mathcal{V}_j| > 1$? So far we have not yet used the information of moment matrix $M_d^{\Delta_i}(y^*)$. If $M_d^{\Delta_i}(y^*)$ also satisfies the flat extension condition, we can extract the tuples $x_{\Delta_i}^* = (x_k^*)_{k \in \Delta_i}$ from $M_d^{\Delta_i}(y^*)$. Let $\mathcal{X}_{\Delta_i}$ be set of all such tuples that can be extracted from $M_d^{\Delta_i}(y^*)$. One might ask whether $\mathcal{V}_i$ and $\mathcal{X}_{\Delta_i}$ are consistent, that is, does $x_{\Delta_i}^* \in \mathcal{X}_{\Delta_i}$ imply that $x_k^* \in \mathcal{V}_k$ for all $k \in \Delta_i$? Under the flat extension assumption, the answer is yes, which is due to the following theorem.

Theorem 3.9. *Suppose y^* is one optimal solution to (3.6)-(3.8) such that all $M_d^{\Delta_i}(y^*)$ satisfy the flat extension condition. Then for any $x_{\Delta_i}^* \in \mathcal{X}_{\Delta_i}$, it holds that $x_k^* \in \mathcal{V}_k$ for all $k \in \Delta_i$.*

Proof. Let $\mathcal{X}_{\Delta_i} = \{x_{\Delta_i}^{(1)}, x_{\Delta_i}^{(2)}, \cdots, x_{\Delta_i}^{(r)}\}$ be the r-atomic representing support for $\mathcal{M}_d^{\Delta_i}(y^*)$. Then we have decomposition

$$M_d^{\Delta_i}(y^*) = \sum_{\ell=1}^r \lambda_\ell \, \mathbf{m}_2(x_{\Delta_i}^{(\ell)}) \mathbf{m}_2(x_{\Delta_i}^{(\ell)})^T$$

for some $\lambda_1, \cdots, \lambda_r > 0, \sum_{\ell=1}^r \lambda_\ell = 1$. Notice that $M_d^{\delta_k}(y^*)$ is a principle submatrix of $M_d^{\Delta_i}(y^*)$. So we also have that for every $k \in \Delta_i$

$$M_d^{\delta_k}(y^*) = \sum_{\ell=1}^r \lambda_\ell \, \mathbf{m}_2(x_k^{(\ell)}) \mathbf{m}_2(x_k^{(\ell)})^T.$$

This means that $\{x_k^{(1)}, x_k^{(2)}, \cdots, x_k^{(r)}\}$ is a r-atomic representing support for moment matrix $M_d^{\delta_k}(y^*)$ (some $x_k^{(\ell)}$ might be the same). By the definition of $\mathcal{V}_i$, we have $\{x_k^{(1)}, \cdots, x_k^{(r)}\} \subseteq \mathcal{V}_k$. $\square$

Theorem 3.10. *Suppose y^* is one optimal solution to (3.6)-(3.8) such that all $M_d^{\Delta_i}(y^*)$ satisfy the flat extension condition. Then any $x^* = (x_1^*, \cdots, x_n^*)$ with $x_k^* \in \mathcal{V}_k$ and $x_{\Delta_i}^* \in \mathcal{X}_{\Delta_i}$ for all k and i is a global optimal minimizer of $f(x)$.*

Proof. Fix x^* as in the theorem. Since $M_d^{\Delta_i}(y^*)$ satisfies the flat extension condition, we have the decomposition

$$M_d^{\Delta_i}(y^*) = \lambda_{\Delta_i} \mathbf{m}_d(x_{\Delta_i}^*) \mathbf{m}_2(x_{\Delta_i}^*)^T + \hat{M}_{\Delta_i}$$

where $1 \geq \lambda_{\Delta_i} > 0$ and $\hat{M}_{\Delta_i} \succeq 0$. Now let $\lambda = \min_i \lambda_{\Delta_i} > 0$ and

$$M_{\Delta_i} = (\lambda_{\Delta_i} - \lambda) \mathbf{m}_2(x_{\Delta_i}^*) \mathbf{m}_2(x_{\Delta_i}^*)^T + \hat{M}_{\Delta_i} \succeq 0.$$

Notice that $\hat{M}_{\Delta_i}$ and M_{Δ_i} are also moment matrices. Without loss of generality, we can assume $\lambda < 1$, since otherwise each $M_2^{\Delta_i}(y^*)$ has rank one and then x^* is obviously a global minimizer. For every $\alpha \in \mathcal{F}_i$, define $\hat{y}_\alpha = \left(x_{\Delta_i}^*\right)^\alpha$ and $\hat{y} = (\hat{y}_\alpha)_{\alpha \in \mathcal{F}}$. Let $\tilde{y} = (\tilde{y}_\alpha)_{\alpha \in \mathcal{F}}$ be the vector such that $y^* = \lambda \hat{y} + (1 - \lambda)\tilde{y}$. Then it holds

$$M_d^{\Delta_i}(y^*) = \lambda M_d^{\Delta_i}(\hat{y}) + (1 - \lambda) M_d^{\Delta_i}(\tilde{y}).$$

Obviously vector $\tilde{y}$ is feasible for (3.7)-(3.8) since

$$M_d^{\Delta_i}(\tilde{y}) = \frac{1}{1 - \lambda}\left(M_d^{\Delta_i}(y^*) - \lambda M_d^{\Delta_i}(\hat{y})\right) = \frac{1}{1 - \lambda} M_{\Delta_i} \succeq 0.$$

Since y^* is optimal, we can see $\sum_{\alpha \in \mathcal{F}} f_\alpha y_\alpha^* \leq \sum_{\alpha \in \mathcal{F}} f_\alpha \hat{y}_\alpha$ and $\sum_{\alpha \in \mathcal{F}} f_\alpha y_\alpha^* \leq \sum_{\alpha \in \mathcal{F}} f_\alpha \tilde{y}_\alpha$. By linearity, it holds

$$f_\Delta^* = \sum_{\alpha \in \mathcal{F}} f_\alpha y_\alpha^* = \lambda \sum_{\alpha \in \mathcal{F}} f_\alpha \hat{y}_\alpha + (1 - \lambda) \sum_{\alpha \in \mathcal{F}} f_\alpha \tilde{y}_\alpha.$$

Therefore, we must have $\sum_{\alpha \in \mathcal{F}} f_\alpha \hat{y}_\alpha = f_\Delta^*$ since $0 < \lambda < 1$. On the other hand, by the definition of $\hat{y}$, we know $f(x^*) = \sum_{\alpha \in \mathcal{F}} f_\alpha \hat{y}_\alpha = f_\Delta^*$. Thus x^* is one point at which the polynomial $f(x)$ attains its lower bound f_Δ^*, which implies that x^* is a global minimizer of $f(x^*)$. $\square$

The algorithm for minimizing $f(x)$ via sparse SOS relaxation (3.1)-(3.3) is as follows.

Algorithm 3.11 (Minimizing sum of polynomials)**.**

Input: $n, m, \Delta_i, f_i(x_{\Delta_i})\, (i = 1, \cdots, m)$

Output: $\mathcal{V}_i$ and $\mathcal{X}_{\Delta_i}\, (i = 1, \cdots, m)$

Begin

    Step 1: Solve the dual problem (3.6)-(3.8). Get the optimal solution y^*.

    Step 2: For each $1 \leq k \leq n$, find the set $\mathcal{V}_k$ of points that can be extracted from $M_d^{\delta_k}(y^*)$.

    Step 3: For every k with $|\mathcal{V}_k| > 1$, find the set $\mathcal{X}_{\Delta_i}$ from $M_d^{\Delta_i}(y^*)$ whenever $k \in \Delta_i$.

End

As an example, let us illustrate how to solve the global optimization problem

$$\min_{x \in \mathbb{R}^3} \quad \underbrace{(x_1^2 - 1)^2 + (x_1 - x_2)^4}_{f_1(x_{\Delta_1})} + \underbrace{(x_2 - x_3)^4}_{f_2(x_{\Delta_2})}$$

and find global minimizers. Here $\Delta_1 = \{1, 2\}$ and $\Delta_2 = \{2, 3\}$. Solve the dual problem (3.6)-(3.8) and we get solutions

$$\mathcal{M}_1^{\Delta_1}(y^*) = \mathcal{M}_1^{\Delta_2}(y^*) = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}.$$

Both $\mathcal{M}_1^{\Delta_1}(y^*)$ and $\mathcal{M}_1^{\Delta_2}(y^*)$ have rank two and satisfy the flat extension condition. Using the technique from [12], we can extract

$$\mathcal{V}_1 = \mathcal{V}_2 = \mathcal{V}_3 = \{-1, 1\}$$

and

$$\mathcal{X}_{\Delta_1} = \left\{ \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\}, \quad \mathcal{X}_{\Delta_2} = \left\{ \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\}.$$

Since the x_2-component from $\mathcal{X}_{\Delta_1}$ and $\mathcal{X}_{\Delta_2}$ must be the same, we know there are two global minimizers $x^* = \pm(1, 1, 1)$.

3.4. Nonlinear least squares problems

Now we consider the special case that each $f_i(x_{\Delta_i})$ is a square of some polynomial, say, $f_i(x_{\Delta_i}) = g_i^2(x_{\Delta_i})$. Then the global minimization of $f(x) = \sum_i f_i(x_{\Delta_i})$ is equivalent to solving the nonlinear least squares (NLS) problem associated with the polynomial system:

$$g_1(x_{\Delta_1}) = g_2(x_{\Delta_2}) = \cdots = g_m(x_{\Delta_m}) = 0. \tag{3.10}$$

In this situation, the polynomial function is often nonconvex and it is very difficult for general numerical optimization schemes like branch-bound to find the global minimizer of $f(x)$.

Theorem 3.12. *If the polynomial system (3.10) admits a solution, then the sparse SOS relaxation (3.1)-(3.3) is exact, i.e., $f_\Delta^* = f_{sos}^* = f^*$.*

Proof. Obviously $f^* = 0$. And $\gamma = 0$ is a feasible solution to problem (3.1)-(3.3), since $f(x)$ itself is a sparse SOS representation as in (3.2)-(3.3). So $f_\Delta^* \geq 0$, and hence all the inequalities in the Theorem 3.1 become equalities. $\square$

Remark 3.13. When the polynomial system (3.10) admits a solution, we necessarily have $f^* = 0$. This might be trivial in some sense. However, the optimal solution y^* to the dual problem (3.6)-(3.8) can help recover the real zeros of polynomial system (3.10), which are absolutely the global minimizers of $f(x)$. See the example below.

Example 3.14. Consider the sparse polynomial system

$$2x_1^2 - 3x_1 + 2x_2 - 1 = 0$$
$$2x_i^2 + x_{i-1} - 3x_i + 2x_{i+1} - 1 = 0 \, (i = 2, \cdots, n-1)$$
$$2x_n^2 + x_{n-1} - 3x_n - 1 = 0.$$

This polynomial system is consistent and has at least two real solutions. Set $n = 20$. We apply sparse SOS relaxation (3.1)-(3.3) to solve the least squares problem and get the lower bound $f_\Delta^* \approx -2.0 \cdot 10^{-11}$. Using the optimal dual solution, we obtain two real solutions (only the first four digits are shown)

$$\hat{x} = (1.8327, -0.1097, -0.5929, -0.6860, -0.7032, -0.7064, -0.7070, -0.7071, -0.7071, -0.7071,$$
$$-0.7071, -0.7070, -0.7068, -0.7064, -0.7051, -0.7015, -0.6919, -0.6658, -0.5960, -0.4164)$$
$$\tilde{x} = (-0.5708, -0.6819, -0.7025, -0.7063, -0.7070, -0.7071, -0.7071, -0.7071, -0.7071, -0.7071,$$
$$-0.7071, -0.7070, -0.7068, -0.7064, -0.7051, -0.7015, -0.6919, -0.6658, -0.5960, -0.4164).$$

3.5. A sparser SOS relaxation

From Theorem 3.12, we know the sparse SOS relaxation (3.1)-(3.3) is exact whenever the polynomial system (3.10) admits a solution, and the optimal dual solution can help recover the

real zeros. This fact makes it possible to exploit the sparsity of each $f_i(x_{\Delta_i})$ further. In (3.1)-(3.3), we assume each $f_i(x_{\Delta_i})$ is a dense polynomial. However, if each $f_i(x_{\Delta_i})$ is sparse, we can get a sparser SOS relaxation. It is obvious that

$$\operatorname{supp}(f_i) \subseteq \mathcal{G}_i + \mathcal{G}_i$$

where $\mathcal{G}_i$ is the convex hull of $\{\alpha \in \mathbb{N}^n : 2\alpha \in \operatorname{supp}(f_i)\}$. This motivates us to propose the sparser SOS relaxation

$$f_{\Delta_s}^* := \max \quad \gamma \tag{3.11}$$

$$s.t. \quad f(x) - \gamma = \sum_{i=1}^m \mathbf{m}_{\mathcal{G}_i}(x)^T W_i \mathbf{m}_{\mathcal{G}_i}(x) \tag{3.12}$$

$$W_i \succeq 0, \ i = 1, \cdots, m. \tag{3.13}$$

Here $\mathbf{m}_{\mathcal{G}_i}(x_{\Delta_i})$ is the column vector of all monomials in x with exponents from $\mathcal{G}_i$. The size of matrix W_i is equal to the cardinality of $\mathcal{G}_i$. Similar to (3.1)-(3.3), the dual of (3.11)-(3.13) can be derived to be

$$f_{\Sigma_s}^* := \min \quad \sum_{\alpha} f_\alpha y_\alpha \tag{3.14}$$

$$s.t. \quad \mathcal{M}_{\mathcal{G}_i}(y) \succeq 0, \ i = 1, \cdots, m \tag{3.15}$$

$$y_0 = 1. \tag{3.16}$$

Here the sparse moment matrix $\mathcal{M}_{\mathcal{G}_i}(y)$ is indexed by vectors from $\mathcal{G}_i$ and defined as

$$\mathcal{M}_{\mathcal{G}_i}(y)(\alpha, \beta) = y_{\alpha+\beta}$$

for all $\alpha, \beta \in \mathcal{G}_i$.

Theorem 3.15. *The optimal values* $f_{\Sigma_s}^*$, $f_{\Delta_s}^*$, f_Σ^*, f_Δ^*, f_{sos}^*, f^* *satisfy the relationship*

$$f_{\Sigma_s}^* = f_{\Delta_s}^* \leq f_\Sigma^* = f_\Delta^* \leq f_{sos}^* \leq f^*.$$

Proof. Applying the standard duality theory in convex programming as in the proof of Theorem 3.1, we can get the first equality from the left by proving (3.12)-(3.13) has a strict interior point. Since the relaxation (3.11)-(3.13) is a special case of (3.1)-(3.3), we obtain the first inequality from the left. The other relations follow Theorem 3.1. $\square$

Theorem 3.16. *Suppose* $f_i(x_{\Delta_i}) = g_i^2(x_{\Delta_i})$. *If the polynomial system (3.10) admits a solution, then the sparse SOS relaxation (3.11)-(3.13) is exact, i.e.,* $f_{\Delta_s}^* = f_{sos}^* = f^*$.

Proof. The proof is almost the same as for Theorem 3.12. Obviously $f^* = 0$. And $\gamma = 0$ is a feasible solution, since $f(x)$ itself is a sparse SOS representation as in (3.12)-(3.13). So $f_{\Delta_s}^* \geq 0$, and hence all the inequalities in the Theorem 3.15 become equalities. $\square$

Remark 3.17. When the polynomial system (3.10) admits a solution, we must have $f^* = 0$. This lower bound itself might not be interesting. However, the optimal dual solution y^* to (3.14)-(3.16) can help recover the real zeros of polynomial system (3.10), which are absolutely the global minimizers of $f(x)$. This observation is very important and has many applications. See examples in Subsection 5.1.

4 Numerical examples

In this section, we present some numerical experiments using sparse SOS relaxations (3.1)-(3.3) and (3.11)-(3.13). First, we use them to solve some test problems from unconstrained optimization. Second, we generate various random polynomials, test the performance of these sparse SOS relaxations and compare with other methods. All the computations are implemented on a Linux machine with 0.98 GB memory and 1.46 GHz CPU. The SOS relaxations are solved by softwares

SeDuMi [32] and *YALMIP* [20]. Throughout this section, the computation time is in CPU seconds. The accuracy of relaxations is measured by $\frac{|f(\hat{x})-\hat{f}|}{\max\{1,|f(\hat{x})|\}}$, where $\hat{x}$ is one extracted solution and $\hat{f}$ is the computed lower bound.

4.1. Some global optimization test problems

In this subsection, we apply SOS relaxations (3.1)-(3.3) and (3.11)-(3.13) to solve some global optimization test problems from [7, 21, 22]. The relaxation (3.1)-(3.3) is usually applied when each $f_i(\Delta_i)$ is almost dense, and the sparser relaxation (3.11)-(3.13) is usually applied when each $f_i(\Delta_i)$ is sparse. All the test functions in this subsection have global minimum $f^* = 0$. So we use the absolute value of the lower bounds f_Δ^* or $f_{\Delta_s}^*$ to measure the accuracy of the relaxation.

First, consider the following test functions.

- The chained singular function [7]:

$$f(x) = \frac{1}{10^5} \sum_{i \in J} \left((x_i + 10x_{i+1})^2 + 5(x_{i+2} - x_{i+3})^2 + (x_{i+1} - 2x_{i+2})^4 + 10(x_i - 10x_{i+3})^4 \right)$$

where $J = \{1, 3, 5, \cdots, n-3\}$ and n is a multiple of 4. The factor $\frac{1}{10^5}$ is used to scale the coefficients to avoid numerical troubles.

- The chained wood function [7]

$$f(x) = \sum_{i \in J} (100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2 + 90(x_{i+3} - x_{i+2}^2)^2 +$$
$$(1 - x_{i+2})^2 + 10(x_{i+1} + x_{i+3} - 2)^2 + 0.1(x_{i+1} - x_{i+3})^2)$$

where $J = \{1, 3, 5, \cdots, n-3\}$ and n is a multiple of 4.

- The generalized Rosenbrock function [22]:

$$f(x) = \sum_{i=2}^{n} \left\{ 100(x_i - x_{i-1}^2)^2 + (1 - x_i)^2 \right\}.$$

We apply SOS relaxation (3.1)-(3.3) to minimize these polynomial functions. The relaxation (3.1)-(3.3) is solved by softwares *SeDuMi* and *YALMIP*. The accuracy and consumed CPU time are in Table 1. The problems are solved from size 100 to 500. For these polynomials, the relaxation (3.1)-

n	chained singular		chained wood		gen. Rosen.	
	accu.	time	accu.	time	accu.	time
100	3.2e-09	2.72	3.5e-10	1.52	9.0e-8	0.95
200	3.0e-10	5.29	3.7e-10	2.25	1.8e-7	1.46
300	5.0e-09	8.01	3.8e-10	3.19	2.7e-7	2.24
400	5.0e-10	11.64	3.9e-10	4.12	3.6e-7	2.88
500	4.9e-09	33.09	3.9e-10	5.12	4.5e-7	3.45

Table 1: The performance of sparse SOS relaxation (3.1)-(3.3)

(3.3) is almost the same as the one in [34]. This is because the csp graphs of these polynomials are chordal graphs. However, if the csp graphs are sparse but their chordal extensions are much dense, then the relaxation in [34] is very similar to the dense SOS relaxation. In such situations, the relaxation (3.1)-(3.3) might be more suitable. For example, to minimize the sparse polynomial

$$(x_1^2 + x_2^2 - 1)^2 + (x_2^2 + x_3^2 - 1)^2 + \cdots + (x_{n-1}^2 + x_n^2 - 1)^2 + (x_n^2 + x_1^2 - 1)^2,$$

the chordal extension of the csp graph is the complete graph, and hence the sparse SOS relaxation using chordal extension is the same as the dense SOS relaxation. However, the sparse relaxation (3.1)-(3.3) is very suitable for this problem.

Second, consider the following test functions.

13

- Broyden tridiagonal function [21]:

$$f(x) = \sum_{i=1}^{n} \left((3 - 2x_i)x_i - x_{i-1} - 2x_{i+1} + 1 \right)^2$$

where $x_0 = x_{n+1} = 0$.
- Broyden banded function [21]:

$$f(x) = \sum_{i=1}^{n} \left(x_i(2 + 10x_i^2) + 1 - \sum_{j \in J_i} (1 + x_j)x_j \right)^2$$

where $J_i = \{j : j \neq i, \max(1, i-5) \leq j \leq \min(n, i+1)\}$.
- Discrete boundary value function [7]:

$$f(x) = \sum_{i=1}^{n} \left(2x_i - x_{i-1} - x_{i+1} + \frac{1}{2} h^2 (x_i + t_i + 1)^3 \right)^2$$

where $h = \frac{1}{n+1}$, $t_i = ih$, $x_0 = x_{n+1} = 0$.

These three polynomials have sparse summand polynomial $f_{\Delta_i}^*$. So we apply the sparser SOS relaxation (3.11)-(3.16) and solve it by using softwares *SeDuMi* and *YALMIP*. The computational results are in Table 2. All the problems are solved quite well in a few seconds.

Broyden Tridiagonal			Broyden banded			disc. bound val.		
n	accu.	time	n	accu.	time	n	accu.	time
100	1.2e-7	2.65	10	3.6e-11	9.72	10	6.0e-12	0.92
200	2.3e-7	2.69	15	2.2e-10	17.28	20	3.4e-11	1.57
300	5.0e-7	3.58	20	1.6e-10	25.27	25	1.6e-11	2.28
400	3.0e-6	4.53	25	1.8e-10	35.19	30	1.1e-11	2.47
500	4.1e-6	5.44	30	4.9e-10	45.30	35	3.9e-11	3.00

Table 2: The performance of sparse SOS relaxation (3.11)-(3.13)

For the Broyden tridiagonal function, we can also apply the sparse relaxation (3.1)-(3.3) or chordal extension from [34]. They are slightly more expensive. For $n = 500$, the problem can be solved in about ten seconds with similar accuracy. However, for the Broyden banded function and discrete boundary value function, the relaxation (3.1)-(3.3) and the method in [34] are much more expensive. For instance, when n has values 10 or bigger, they are usually difficult to implement due to computer memory restrictions.

One interesting observation in Table 2 is that the accuracy for the Broyden tridiagonal function is not as high as for the other two functions. One possible reason is that the global minimizer of Broyden tridiagonal function is not unique and there are additional numerical troubles caused from extracting minimizers. This illustrates that the computation is more numerically difficult when there are multiple global solutions.

4.2. Randomly generated test problems

In this subsection, we present the computational results for randomly generated polynomials. The aim is to test the performance of the sparse SOS relaxation (3.1)-(3.3) for minimizing random polynomials and compare with other sparse SOS methods. For these randomly generated polynomials, solve the sparse relaxation (3.1)-(3.3) by using softwares *SeDuMi* and *YALMIP*. Then we get lower bounds f_{Δ}^* and extract minimizers $\hat{x}$. Since we do not know the true global minimizers in advance, the accuracy of $\hat{x}$ can be measured by $err = \frac{|f(\hat{x}) - f_{\Delta}^*|}{\max\{1, |f(\hat{x})|\}}$. The smaller err is, the more accurate $\hat{x}$ is, since f_{Δ}^* is a guaranteed lower bound.

4.2.1 Randomly generated sums of small polynomials

In this subsubsection, we randomly generate sparse polynomials $f(x)$ of the form (1.1) and use them to test the performance of the sparse relaxation (3.1)-(3.3). Then the csp graph of $f(x)$ is usually not chordal and its chordal extension is often much less sparse. So the method in [34] is usually expensive for these polynomials. We let $m = n$ and choose f_i to have form

$$f_i(x_{\Delta_i}) = \mathbf{m}_d(x_{\Delta_i})^T \cdot A_i \cdot \mathbf{m}_d(x_{\Delta_i}) + b_i^T \mathbf{m}_{2d-1}(x_{\Delta_i})$$

where Δ_i are chosen to be random subsets of $[n]$ with cardinality at most $\|\Delta\|$. Here $N_i = \binom{|\Delta_i|+d}{d}$, $A_i = nI_{N_i} + BB^T$, $B \in \mathbb{R}^{N_i \times N_i}$ and $b_i \in \mathbb{R}^{\binom{|\Delta_i|+d-1}{d-1}}$ are random. So each A_i is positive definite. This choice guarantees that the global minimizers of $f(x)$ are contained in some compact set.

| | $\|\Delta\| = 3$ | | | | $\|\Delta\| = 4$ | | | |
| | CPU seconds | | | accu | CPU seconds | | | accu |
n	max	avr.	min	max	max	avr.	min	max
20	0.85	0.62	0.54	4.1e-9	1.46	1.15	0.91	2.4e-9
40	1.22	1.07	0.91	1.9e-9	2.86	2.49	2.25	2.9e-9
60	1.80	1.55	1.45	2.9e-9	4.43	4.17	3.91	3.1e-9
80	2.30	2.18	2.02	2.3e-9	6.26	5.94	5.24	3.7e-9
100	3.02	2.70	2.33	2.8e-9	7.85	7.41	7.01	5.0e-9

Table 3: Computational results for quartic polynomials with different sizes

First, let $2d = 4$ and n be $20, 40, 60, 80, 100$. For each $\|\Delta\|$ (3 or 4) and n, we generate 100 random polynomials in the way described above. For each one, we solve the sparse SOS relaxation (3.1)-(3.3) by using softwares *SeDuMi* and *YALMIP*, and get the lower bound f_Δ^* and optimal dual solution $\hat{y}$. For all these randomly generated polynomials, the moment matrices $\mathcal{M}_d^{\Delta_i}(\hat{y})$ have numerical rank one. So we can easily extract the minimizer $\hat{x}$. The maximum, average and minimum of consumed CPU time are in Table 3. For these random polynomials, we just record the maximum error of the extracted minimizers. From Table 3, for $\Delta = 4$, we can find the global minimizer of a quartic sparse polynomial of 100 variables with error $O(10^{-9})$ within about 8 CPU seconds.

| | $\|\Delta\| = 3$ | | | | $\|\Delta\| = 4$ | | | |
| | CPU seconds | | | err | CPU seconds | | | err |
$2d$	max	avr	min	max	max	avr	min	max
4	1.01	0.87	0.77	2.5e-9	2.33	1.93	1.65	2.4e-9
6	3.22	2.96	2.67	1.8e-9	17.16	14.92	11.71	2.2e-9
8	13.07	11.44	10.13	1.7e-8	136.67	119.90	107.28	9.4e-8

Table 4: Computational results for polynomials of size $n = 30$ with different degrees

Second, let $n = 30$ and $2d$ be $4, 6, 8$. For each $\|\Delta\|$ (3 or 4) and $2d$, we generate 100 random polynomials in the way described in the above. For each one, solve the sparse SOS relaxation (3.1)-(3.3) by using softwares *SeDuMi* and *YALMIP*, and get the lower bounds f_Δ^* and optimal dual solution $\hat{y}$. Similarly, all moment matrices $\mathcal{M}_d^{\Delta_i}(\hat{y})$ have rank one, and the minimizer $\hat{x}$ can be extracted easily. The maximum, average and minimum of the consumed CPU time, and the maximum error of extracted minimizers are in Table 4. For $\|\Delta\| = 4$, the global minimizer of such generated polynomials of degree 8 and 30 variables can be found with error $O(10^{-8})$ within about 120 seconds.

We remark that the sparsity technique in [34] is too expensive to be implementable for minimizing these random polynomials generated in the way as above because of computer memory

limitations. For these random polynomials, the sparsity technique using chordal extension is almost as expensive as the general dense SOS relaxation. This is because the chordal extensions of csp graphs of these polynomials are usually much more dense than the original csp graphs. However, as we have seen in the above, the SOS relaxation (3.1)-(3.3) is very suitable for these polynomials.

4.2.2 Random sparse polynomials with given chordal extension

In this subsubsection, we generate random sparse polynomials in a similar way as in [34], and compare the performance of our sparse SOS relaxation (3.1)-(3.3) with the one in [34] using chordal extension. Generate a chordal graph randomly such that the size of every maximal clique is at most 6. Let $\{C_1, \cdots, C_m\}$ be the set of maximal cliques. If we choose $\Delta_i = C_i$, then the sparse SOS relaxation (3.1)-(3.3) is the same as the one using chordal extension. Therefore, to make a reasonable comparison, for each C_i, we choose a random subset $\Delta_i \subseteq C_i$. Choose each small polynomial f_i to have the form

$$f_i(x_{\Delta_i}) = \mathbf{m}_d(x_{\Delta_i})^T \cdot A_i \cdot \mathbf{m}_d(x_{\Delta_i}) + b_i^T \mathbf{m}_{2d-1}(x_{\Delta_i}).$$

Here $N_i = \binom{|\Delta_i|+d}{d}$, $A_i = nI_{N_i} + BB^T$, $B \in \mathbb{R}^{N_i \times N_i}$ and $b_i \in \mathbb{R}^{\binom{|\Delta_i|+d-1}{d-1}}$ are random. The global minimizers of $f(x) = \sum_i f_i(x_{\Delta_i})$ generated as above always exist and are contained in some compact set.

| | relaxation (3.1)-(3.3) | | | | relaxation using chordal extension | | | |
| | CPU seconds | | | accu | CPU seconds | | | accu |
n	max	avr.	min	max	max	avr.	min	max
20	1.75	1.21	0.96	6.8e-9	2.15	1.78	1.43	5.5e-9
40	3.07	2.69	2.24	7.5e-9	4.08	3.51	3.12	4.9e-9
60	4.99	4.54	3.82	6.7e-9	7.88	6.93	5.65	6.4e-9
80	6.59	5.87	5.23	6.3e-9	10.84	9.57	8.59	5.7e-9
100	9.34	7.64	7.11	7.2e-9	13.45	12.76	11.74	4.3e-9

Table 5: Comparison with chordal extension on quartic polynomials

For polynomials randomly generated as above, the technique in [34] using chordal extension is a good choice, because there exists one sparse chordal extension of the csp graph. Now we compare the computational results for these two methods.

| | relaxation (3.1)-(3.3) | | | | relaxation using chordal extension | | | |
| | CPU seconds | | | accu | CPU seconds | | | accu |
$2d$	max	avr.	min	max	max	avr.	min	max
4	2.87	1.98	1.35	7.2e-9	3.06	2.21	1.69	4.3e-9
6	22.61	16.78	10.53	6.9e-9	32.15	23.91	13.51	5.1e-9
8	193.45	131.17	98.75	6.7e-9	253.79	186.84	112.37	5.8e-9

Table 6: Comparison with chordal extension on polynomials with 30 variables

First, let $2d = 4$ and n be $20, 40, 60, 80, 100$. For each n, generate 50 random polynomials as above. For each of these random polynomials, solve the relaxation (3.1)-(3.3), find a chordal extension of the csp graph of $f(x)$ and then apply the sparse relaxation in [34]. Both relaxations are solved by softwares *SeDuMi* and *YALMIP*. Then we extract minimizers $\hat{x}$ from moment matrices. The computational results are in Table 5. For these solved problems, we just record the maximum error of the relaxation. Second, let $n = 30$ and $2d$ be $4, 6, 8$. For each $2d$, generate

50 random polynomials as above. For each one, solve the problem by the relaxation (3.1)-(3.8) and the one in [34] using chordal extension. They are solved by softwares *SeDuMi* and *YALMIP*. The computational results are in Table 6.

From Tables 5 and 6, we observe that for polynomials randomly generated as above the sparse SOS relaxation (3.1)-(3.3) is slightly more computationally efficient than the one using chordal extension. As we can see, for these random polynomials, there is not much difference between the qualities of these two kinds of sparse SOS relaxations. The distinction between their qualities depends on specific problems. Of course, theoretically the sparse relaxation using chordal extension in [34] is at least as tight as the relaxation (3.1)-(3.3).

4.2.3 Random dense polynomials

In this subsubsection, we test the performance of our sparse SOS relaxation on minimizing general dense polynomials. We observe that every polynomial $f(x)$ is a summation of monomials whose number of variables is at most the degree $\deg(f)$. So the sparse SOS relaxation (3.1)-(3.3) is attractive when the degree $2d$ is small like 4. We generate the random dense polynomials as follows

$$f(x) = \mathbf{m}_d(x)^T \cdot A \cdot \mathbf{m}_d(x) + b^T \mathbf{m}_{2d-1}(x).$$

Here $N = \binom{n+d}{d}$, $A = nI_N + BB^T$, $B \in \mathbb{R}^{N \times N}$ is a random matrix and $b \in \mathbb{R}^{\binom{n+2d-1}{n}}$ is a random vector. So the global minimizers of $f(x)$ generated this way are contained in some compact set. Note that $f(x)$ is also a summation of small polynomials. Let Δ_i be the subsets of $[n]$ with cardinality $2d$. Then we can write

$$f(x) = \sum_{i=1}^{\binom{n}{2d}} f_i(x_{\Delta_i})$$

for some small polynomials $f_i(x_{\Delta_i})$.

n	16	17	18	19	20	21	22	23
max time	335.29	569.74	901.32	1505.45	2249.19	3257.86	4734.25	7060.72
avr. time	241.48	455.32	751.69	1245.22	2070.70	2989.45	4497.84	6419.53
min time	205.60	397.11	688.58	1052.70	1893.02	2676.62	4197.95	5874.28
accuracy	7.3e-9	6.7e-9	7.4e-9	6.9e-9	8.1e-9	6.5e-9	7.9e-9	8.5e-9

Table 7: Computational results for dense quartic polynomials

Since $\|\Delta\| = 2d$, which should not be big for the effectiveness of the sparse relaxation (3.1)-(3.3), we test for the case that $2d = 4$. Let n be $16, 17, 18, 19, 20, 21, 22, 23$. For each pair (n, d) of these values, generate 50 random examples as above. For each random polynomial, solve the sparse relaxation (3.1)-(3.3) by using softwares *SeDuMi* and *YALMIP*. The consumed CPU time and the accuracy of relaxation are in Table 7. We can see that the obtained solutions are very good within reasonably acceptable time. When $n \geq 24$, the sparse relaxation (3.1)-(3.3) is then also too expensive to be implementable due to computer memory restrictions.

For these randomly generated dense polynomials, the general dense SOS relaxation and sparse SOS relaxations like in [34] are not implementable for $n \geq 16$, due to either computer memory shortage or the consumed time being more than 10 hours. However, when $2d$ is small like 4, the sparse SOS relaxation (3.1)-(3.3) can solve bigger dense polynomial optimization problems which can not be solvable by other methods.

5 Applications

Minimizing a summation of small polynomials arises in various applications. Many big polynomials in applications often come in this form. In such situations, the sparse SOS relaxation

(3.1)-(3.3) or (3.11)-(3.13) is very useful. In this section, we show some applications in solving sparse polynomial systems and sensor network localization.

5.1. Solving sparse polynomial system

Suppose we are trying to solve the sparse polynomial system

$$g_1(x_{\Delta_1}) = 0, \ g_2(x_{\Delta_2}) = 0, \ \cdots, \ g_m(x_{\Delta_m}) = 0.$$

In some applications, these equations are redundant or even inconsistent. When the polynomial system does not admit a solution, we want to seek a least squares solution, which is often useful in applications.

This problem can be formulated as finding the global minimizer of the sparse polynomial

$$f^* := \min_{x \in \mathbb{R}^n} \ f(x) = \sum_{i=1}^m g_i^2(x_{\Delta_i}).$$

The polynomial system has a real zero if and only if $f^* = 0$. When $f^* = 0$, the global minimizers are precisely the real zeros of the polynomial system. When $f^* > 0$, the global minimizers are the least squares solutions.

One important sparse polynomial system of the above form is from computing the numerical solutions of nonlinear differential equations. Consider the two-point boundary value problem (BVP)

$$F(t, x, x', x'') = 0, \ x(a) = \alpha, \ x(b) = \beta$$

where $F(t, x, x', x'')$ is polynomial function in t, x, x', x''. To find the numerical solution, the central difference approximation with a uniform mesh is often used to discretize the derivatives. Let N be a positive integer and set $h = \frac{b-a}{N+1}$. Then we get polynomial difference equations

$$F\left(t_k, x_k, \frac{x_{k+1} - x_{k-1}}{2h}, \frac{x_{k-1} - 2x_k + x_{k+1}}{h^2}\right) = 0, \ k = 1, \cdots, N$$

where $x_0 = \alpha$, $x_{N+1} = \beta$ and $t_k = a + hk$. Every polynomial on the left involves 2 or 3 variables x_{k-1}, x_k, x_{k+1}. So this is a sparse polynomial system. There are several methods for solving this kind of polynomial system, like Newton's method and homotopy methods. Newton's method is very fast, but often require an accurate initial guess. Homotopy methods do not require a "satisfactory" guess and work well for small N, but are expensive to implement for large N. We refer to [1] and the references therein for work in this area. When N is large, this polynomial system is large but sparse. We solve this system by applying the sparse SOS relaxation (3.11)-(3.16) for big N (up to 100 or even bigger).

Example 5.1 ([1])**.** Consider a basic BVP

$$x'' - 2x^3 = 0, \ x(0) = \frac{1}{2}, \ x(1) = \frac{1}{3}.$$

The exact solution to this problem is $x(t) = \frac{1}{t+2}$. Now we discretize the differential equation with mesh size $h = \frac{1}{N+1}$, then get the difference equation

$$\frac{1}{2} - 2x_1 + x_2 - 2h^2 x_1^3 = 0,$$
$$x_{k-1} - 2x_k + x_{k+1} - 2h^2 x_k^3 = 0, \ k = 2, \cdots, N-1$$
$$x_{N-1} - 2x_N + \frac{1}{3} - 2h^2 x_N^3 = 0.$$

This is a polynomial system about $x_1, x_2, \cdots, x_N$. We can solve this polynomial system as a nonlinear least squares problem by applying sparse SOS relaxation (3.11)-(3.16). The computational results are in Table 8. The equation error is defined to be the infinity norm of the residuals

18

N	eqn. error	$\|x_k - x(t_k)\|_\infty$	$\|x_k - x(t_k)\|_\infty/h^2$	time
5	2.8937e-07	7.0252e-05	2.5291e-003	0.52
10	2.3329e-07	1.9570e-05	2.3680e-003	0.77
20	5.2879e-07	1.5041e-05	6.6331e-003	1.18
30	2.6194e-07	1.9413e-05	1.8656e-002	2.09
40	3.0304e-07	4.3344e-05	7.2861e-002	3.99
50	6.5375e-07	1.5124e-04	3.9338e-001	6.82
60	1.5271e-06	4.8695e-04	1.8119e+00	7.77
70	1.2555e-06	5.2428e-04	2.6429e+00	9.16
80	9.7315e-07	6.1330e-04	4.0239e+00	9.78
90	2.7519e-06	1.9311e-03	1.5991e+01	10.81
100	1.8628e-06	8.1425e-04	8.3062e+00	8.79

Table 8: The performance of (3.1)-(3.8) solving the equations in Example 5.1

of the left hand side of the polynomial system, which measures the quality of how the polynomial systems are solved. The obtained solutions have equation error from $\mathcal{O}(10^{-6})$ to $\mathcal{O}(10^{-7})$. If we want to make them more accurate, they can be used as the initial guesses in Newton's methods for refining. The accuracy of the discretization is defined to be the difference between computed solution x_k and true solution $x(t_k) = \frac{1}{2+t_k}$ where $t_k = \frac{k}{N+1}$. Since the discretization has error $\mathcal{O}(h^2)$, we expect that $\|x_k - x(t_k)\|_\infty/h^2$ is a constant. When $N \leq 40$, we can see that $\|x_k - x(t_k)\|_\infty/h^2$ is almost constant. When $N \geq 50$, $SeDuMi$ experienced numerical troubles, and the returned solutions are not as accurate as for the smaller Ns. This explains why $\|x_k - x(t_k)\|_\infty$ and $\|x_k - x(t_k)\|_\infty/h^2$ becomes bigger for $N \geq 50$. Time records the CPU seconds consumed by the SDP solver $SeDuMi$. For $N = 100$, the computation takes less CPU time than for $N = 80$ or $N = 90$. This is because the numerical troubles make $SeDuMi$ terminate earlier.

Example 5.2. Consider another BVP

$$x'' + \frac{1}{2}(x + t)^3 = 0, \ \ x(0) = 0, \ x(1) = 0.$$

Now we discretize the differential equation with mesh size $h = \frac{1}{N+1}$, then get the difference equation

$$2x_1 - x_2 + \frac{1}{2}h^2(x_1 + t_1)^3 = 0$$
$$2x_i - x_{i-1} - x_{i+1} + \frac{1}{2}h^2(x_i + t_i)^3 = 0, \ i = 2, \cdots, N-1$$
$$2x_N - x_{N-1} + \frac{1}{2}h^2(x_N + t_N)^3 = 0$$

This is a polynomial system about $x_1, x_2, \cdots, x_N$. We can solve this polynomial system as a nonlinear least squares problem by applying sparse SOS relaxation (3.11)-(3.16). When $N = 30$, we get the following real solution within about 2.5 CPU seconds (only the first four digits are shown):

$(-0.0159, -0.0312, -0.0459, -0.0600, -0.0735, -0.0864, -0.0985, -0.1099, -0.1205, -0.1302,$
$-0.1391, -0.1470, -0.1540, -0.1599, -0.1646, -0.1682, -0.1705, -0.1715, -0.1710, -0.1689,$
$-0.1651, -0.1596, -0.1521, -0.1425, -0.1307, -0.1164, -0.0995, -0.0796, -0.0567, -0.0302).$

5.2. Sensor Network Localization

The *sensor network location* problem is basically described as follows: find a sequence of unknown vectors $x_1, x_2, \cdots, x_n \in \mathbb{R}^k (k = 1, 2, \cdots)$ (they are called *sensors*) such that distances between these sensors and some other known vectors $a_1, \cdots, a_m$ (they are called *anchors*) are

equal to some given numbers. Now each x_i itself is a k-dimensional vector. To be more specific, let $\mathcal{A} = \{(i,j) \in [n] \times [n] : i < j, \|x_i - x_j\|_2 = d_{ij}\}$, and $\mathcal{B} = \{(i,k) \in [n] \times [m] : \|x_i - a_k\|_2 = e_{ik}\}$, where d_{ij}, e_{ik} are given distances. Then the sensor network localization problem is to find vectors $x_1, x_2, \cdots, x_n$ such that $\|x_i - x_j\|_2 = d_{ij}$ for all $(i,j) \in \mathcal{A}$ and $\|x_i - a_k\|_2 = e_{ik}$ for all $(i,k) \in \mathcal{B}$. Notice that $\mathcal{A}$ and $\mathcal{B}$ only give some partial pairs of distances. $\mathcal{A}$ does not contain all the pairs (i,j) such that $i < j$, and neither does $\mathcal{B}$.

Sensor network localization is also known as the *graph realization problem* or the *distance geometry problem*. Given a graph $G = (V, E)$ along with a real number associated with each edge, graph realization is to assign each vertex a coordinate so that the Euclidean distance between any two adjacent vertices is equal to the real number associated with that edge.

The locations of sensors can be determined from the polynomial system

$$\|x_i - x_j\|_2^2 = d_{ij}^2, \forall (i,j) \in \mathcal{A},$$
$$\|x_i - a_k\|_2^2 = e_{ik}^2, \forall (i,k) \in \mathcal{B}.$$

Usually solving this polynomial system directly is very expensive. Here we solve this polynomial system as a nonlinear least squares problem. Minimize the quartic polynomial function

$$f(x) := \sum_{(i,j) \in \mathcal{A}} \left(\|x_i - x_j\|_2^2 - d_{ij}^2 \right)^2 + \sum_{(i,k) \in \mathcal{B}} \left(\|x_i - a_k\|_2^2 - e_{ik}^2 \right)^2. \tag{5.1}$$

where $x = [x_1, \cdots, x_n]$. x^* is a solution to sensor network localization problem if and only if x^* is a global minimizer of $f(x)$ such that $f(x^*) = 0$. When x^* is a global minimizer such that $f(x^*) > 0$, the distances $d_{i,j}$ and e_{ik} are not consistent, and x^* is a solution in the least squares sense. This polynomial $f(x)$ is of the form (1.1), and our sparse SOS relaxation (3.1)-(3.3) can be applied to solve the problem.

We randomly generate test problems which are similar to those given in [6]. First, we randomly generate $n = 500$ sensor locations $x_1^*, \cdots, x_n^*$ from the unit square $[-0.5, 0.5] \times [-0.5, 0.5]$. The anchors $\{a_1, a_2, a_3, a_4\}$ $(m = 4)$ are chosen to be four fixed points $(\pm 0.45, \pm 0.45)$. Choose edge set $\mathcal{A}$ such that for every sensor x_i^* there are at most 10 sensors x_j^* $(j > i)$ with $(i,j) \in \mathcal{A}$ and $\|x_i^* - x_j^*\|_2 \leq 0.3$. For every $(i,j) \in \mathcal{A}$, compute the distance $\|x_i^* - x_j^*\|_2 = d_{ij}$. Choose edge $\mathcal{B}$ such that every anchor is connected to all the sensors within distance 0.3. For every $(i,k) \in \mathcal{B}$, compute the distance $\|x_i^* - a_k\|_2 = e_{ik}$. Then we apply sparse SOS relaxation (3.1)-(3.8) to minimize polynomial function (5.1). The accuracy of computed sensor locations $\hat{x}_1, \cdots, \hat{x}_n$ will be measured by the Root Mean Square Distance (RMSD) which is defined as RMSD $= \left(\frac{1}{n} \sum_{i=1}^n \|\hat{x}_i - x_i^*\|_2^2 \right)^{\frac{1}{2}}$. We use *SeDuMi* to solve the sparse SOS relaxation (3.1)-(3.8) on a Linux machine with 1.46 GHz CPU and 0.98GB memory. The problem can be solved within about 18 CPU minutes with accuracy $\mathcal{O}(10^{-6})$.

We refer to [24] for more details about sparse SOS methods for sensor network localization.

6 Conclusions and discussions

This paper proposes sparse SOS relaxations for minimizing polynomial functions that are summations of small polynomials. We discuss various properties of these relaxations and the computational issues. We also present applications of this sparsity technique in solving polynomial equations derived from nonlinear differential equations and sensor network localization. As a special case, this sparsity technique provides a heuristic approach to solve bigger dense polynomial optimization problems.

In order to exploit the sparsity, the polynomial and its SOS representation must be sparse. In many applications, the polynomials are often given with sparsity pattern (1.1), and then the sparsity technique proposed in this paper is very suitable. If the sparsity pattern is not given, one important future work is how to represent the polynomial in a sparse pattern such that the technique proposed in this paper is most efficient. Of course, one simple choice is to consider each monomial as a small polynomial.

The idea of this sparse SOS relaxation can be applied in a similar way to solve constrained polynomial optimization problems, provided the objective and constraint polynomials are also

sums of small polynomials. See Kim et al. [14] and Lasserre [18] for related work. To get the global minimum, high order relaxations are usually necessary. Lasserre [18] proved the convergence under the running intersection property. However, unlike the general dense SOS relaxation for minimizing polynomials over compact sets, the convergence might fail when the running intersection property does not hold. As a counterexample, consider the *Minimum Cover Set Problem*. Let $G = (V, E)$ be a graph with vertex set $V = [3]$ and edge set $E = \{(1,2), (1,3), (2,3)\}$. To find the minimum cover set is equivalent to solving

$$\min_{x \in \mathbb{R}^3} \quad f_1(x_{\Delta_1}) + f_2(x_{\Delta_2}) + f_3(x_{\Delta_3})$$
$$s.t. \quad x_1^2 = x_1,\, x_2^2 = x_2,\, x_3^2 = x_3,$$
$$x_1 + x_2 \geq 1,\, x_1 + x_3 \geq 1, x_2 + x_3 \geq 1$$

where $\Delta_1 = \{1,2\}$, $\Delta_2 = \{1,3\}$, $\Delta_3 = \{2,3\}$ and $f_1(x_{\Delta_1}) = \frac{1}{2}(x_1 + x_2)$, $f_2(x_{\Delta_2}) = \frac{1}{2}(x_1 + 1_3)$, $f_3(x_{\Delta_3}) = \frac{1}{2}(x_2 + x_3)$. The running intersection property now fails. However, we can prove that the global minimum $f^* = 2$ and the lower bounds given by sparse SOS relaxations are at most $\frac{3}{2}$. The sparse SOS relaxations do not converge for this example.

Another important future work is to apply the sparse SOS relaxations in solving big real sparse polynomial systems arising from nonlinear differential equations.

References

[1] A.L. Allgower, D.J. Bates, A.J. Sommese and C.W. Wampler. Solution of polynomial systems derived from differential equations. *Computing* **76** (2005), 1-10. `arXiv:math.AG/0402158`.

[2] J. Aspnes, D. Goldberg and Y.R. Yang. On the computational complexity of sensor network localization. *Lecture Notes in Computer Science (3121)*, Springer-Verlag, 2004, pp.32-44.

[3] S. J. Benson and Y.Ye. DSDP3: Dual scaling algorithm for general positive semidefinite programming. *Tech. Report ANL/MCS-P851-1000*, Mathematics and Computer Science Division, Argonne National Laboratory, Feb. 2001.

[4] J. Blair and B. Peyton. An introduction to chordal graphs and clique trees. In J. George, J. Gilbert, and J. Liu, editors, *Graph theory and sparse matrix computations*, pages 1-30, Springer Verlag, 1993.

[5] G. Blekherman. There are significantly more nonnegative polynomials than sums of squares. *Israel J. Math.* 153 (2006), 355–380.

[6] P. Biswas, T.C. Liang, K.C. Toh, T.C. Wang and Y. Ye. Semidefinite Programming Approaches for Sensor Network Localization with Noisy Distance Measurements. To appear in to appear in *IEEE Transactions on Automation Science and Engineering*, 3 (2006), pp. 360–371.

[7] A.R. Conn, N.I.M. Gould and P.L. Toint. Testing a class of methods for solving minimization problems with simple bounds on the variables. *Math. Comp.*, **50** (1988) 399-430.

[8] R.E. Curto and L.A. Fialkow. The truncated complex K-moment problem. *Trans. Amer. Math. Soc.* **352** (2000) 2825–2855.

[9] K. Gatermann and P. Parrilo. Symmetry groups, semidefinite programs, and sums of squares. *Journal of Pure and Appl. Algebra*, Vol. 192, No. 1-3, pp. 95-128, 2004.

[10] B. Grone, C.R. Johnson, E. Marques de Sa, and H. Wolkowicz. Positive definite completions of partial hermitian matrices. *Linear Algebra And Its Applications*, Vol. 58: 109–124, 1984.

[11] D. Henrion and J. Lasserre. GloptiPoly: Global optimization over polynomials with Matlab and SeDuMi. *ACM Trans. Math. Soft.*, 29:165-194, 2003.

[12] D. Henrion and J. Lasserre. Detecting global optimality and extracting solutions in Glop-tiPoly. In *Positive Polynomials in Control*, D. Henrion and A. Garulli, eds., Lecture Notes on Control and Information Sciences, Springer Verlag, 2005.

[13] D. Jibetean and M. Laurent. Semidefinite approximations for global unconstrained polynomial optimization. *SIAM Journal on Optimization*, Vol. 16, No. 2, 490–514, 2005.

[14] S. Kim, M. Kojima and H. Waki. Generalized Lagrangian duals and sums of squares relaxations of sparse polynomial optimization problems. *SIAM Journal on Optimization*, Vol. 15, No. 3, 697-719, 2005.

[15] M. Kojima, S. Kim and H. Waki. Sparsity in Sums of Squares of Polynomials. *Mathematical Programming* Vol. 103 (1) 45-62.

[16] J. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization*, Vol. 11, No. 3, 796–817, 2001.

[17] J. Lasserre. A sum of squares approximation of nonnegative polynomials. *SIAM Journal on Optimization*, Vol. 16, No. 3, 751-765, 2006.

[18] J. Lasserre. Convergent SDP-relaxations in polynomial optimization with sparsity. *SIAM Journal on Optimization*, Vo. 17, No. 3, 822–843, 2006.

[19] M. Laurent. Moment matrices and optimization over polynomials - A survey on selected topics. Preprint, September 2005. `http://homepages.cwi.nl/~monique/`.

[20] J. Löfberg. YALMIP: A MATLAB toolbox for rapid prototyping of optimization problems. `http://control.ee.ethz.ch/~joloef/yalmip.php`.

[21] J. More, B. Garbow and K. Hillstrom. Testing unconstrained optimization software. *ACM Trans. Math. Soft.*, **7** (1981) 17-41.

[22] S.G. Nash. Newton-type minimization via the Lanczos method. *SIAM Journal Numerical Analysis*, **21** (1984), 770-788.

[23] J. Nie, J. Demmel and B. Sturmfels. Minimizing Polynomials via Sum of Squares over the Gradient Ideal. *Mathematical Programming, Series A*, Vol. 106 (2006), No. 3, 587-606.

[24] J. Nie. Sum of squares methods for sensor network localization. Preprint, 2006. *arXiv:math.OC/0605652*.

[25] P. Parrilo and B. Sturmfels. Minimizing polynomial functions, Proceedings of the DIMACS Workshop on Algorithmic and Quantitative Aspects of Real Algebraic Geometry in Mathematics and Computer Science (March 2001), (eds. S. Basu and L. Gonzalez-Vega), American Mathematical Society, 2003, pp. 83–100.

[26] P. Parrilo. Semidefinite Programming relaxations for semialgebraic problems. *Mathematical Programming, Ser. B* **96** (2003), No. 2, 293–320.

[27] P. Parrilo. Exploiting structure in sum of squares programs. *Proceedings for the 42nd IEEE Conference on Decision and Control*, Maui, Hawaii, 2003.

[28] S. Prajna, A. Papachristodoulou, P. Seiler and P. Parrilo. *SOSTOOLS User's Guide*. `http://www.mit.edu/~parrilo/SOSTOOLS/`.

[29] B. Reznick. Extremal psd forms with few terms. *Duke Math. J.*, **45**, 363-374 (1978).

[30] B. Reznick. Some concrete aspects of Hilbert's 17th problem. In *Contemporary Mathematics*, volume 253, pages 251-272. American Mathematical Society, 2000.

[31] M. Schweighofer. Global optimization of polynomials using gradient tentacles and sums of squares. *SIAM Journal on Optimization*, Vol. 17, No. 3, 920-942, 2006.

[32] J.F. Sturm. SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones, *Optimization Methods and Software*, **11&12**(1999)625-653.

[33] B. Sturmfels. Solving systems of polynomial equations. *Amer.Math.Soc., CBMS regional conferences series*, No. 97, providence, Rhode Island, 2002.

[34] H. Waki, S. Kim, M. Kojima and M. Muramatsu. Sums of Squares and Semidefinite Programming Relaxations for Polynomial Optimization Problems with Structured Sparsity. *SIAM Journal on Optimization*, Vol.17 No. 1 218-242 (2006).

[35] H. Waki, S. Kim, M. Kojima and M. Muramatsu. Sparse SDP Relaxation of Polynomial Optimization Problems. `http://www.is.titech.ac.jp/`~`kojima/SparsePOP/index.html`.

[36] H. Wolkowicz, R. Saigal, and L. Vandenberghe, editors. *Handbook of semidefinite programming*. Kluwer's Publisher, 2000.